(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 3 627 367 B1**

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
25.05.2022 Bulletin 2022/21

(21) Application number: **19193452.0**

(22) Date of filing: **23.08.2019**

(51) International Patent Classification (IPC):
G06F 21/57 *(2013.01)* ***G06F 21/55*** *(2013.01)*
H04L 9/30 *(2006.01)* ***H04L 9/32*** *(2006.01)*
H04L 29/06 *(2006.01)*

(52) Cooperative Patent Classification (CPC):
G06F 21/57; G06F 21/556; H04L 9/3073; H04L 9/3218; H04L 9/3234; H04L 9/3247; H04L 63/0421; H04L 2209/42

(54) **SUBVERSION RESILIENT ATTESTATION FOR TRUSTED EXECUTION ENVIRONMENTS**

UNTERVERSIONSBESTÄNDIGE ATTESTATION FÜR VERTRAUENSWÜRDIGE AUSFÜHRUNGSUMGEBUNGEN

ATTESTATION RÉSISTANTE À LA SUBVERSION POUR DES ENVIRONNEMENTS D'EXÉCUTION SÉCURISÉS

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **20.09.2018 US 201862733660 P**
13.08.2019 US 201916538966

(43) Date of publication of application:
25.03.2020 Bulletin 2020/13

(73) Proprietor: **NEC Corporation**
Tokyo 108-0014 (JP)

(72) Inventors:
- **Soriente, Claudio**
  28692 Villafranca del Castillo (ES)
- **Faonio, Antonio**
  28223 Pozuelo de Alarcon, Madrid (ES)
- **Nizzardo, Luca**
  28223 Pozuelo de Alarcon, Madrid (ES)
- **Fiore, Dario**
  28223 Pozuelo de Alarcon, Madrid (ES)

(74) Representative: **Ullrich & Naumann PartG mbB**
Schneidmühlstrasse 21
69115 Heidelberg (DE)

(56) References cited:
WO-A1-2017/049111

- **JAN CAMENISCH ET AL: "Anonymous Attestation with Subverted TPMs", IACR, INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH , vol. 20170628:145434 28 June 2017 (2017-06-28), pages 1-79, XP061023745, Retrieved from the Internet: URL:http://eprint.iacr.org/2017/200.pdf [retrieved on 2017-06-28]**
- **GIUSEPPE ATENIESE ET AL: "Subversion-Resilient Signature Schemes", IACR, INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH, vol. 20150814:120044, 14 August 2015 (2015-08-14), pages 1-41, XP061019027, DOI: 10.1145/2810103.2813635 [retrieved on 2015-08-14]**

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 3 627 367 B1

Description

FIELD

[0001] Among other things, the present application discloses subversion resilient attestation for trusted execution environments.

BACKGROUND

[0002] Remote attestation is a mechanism offered by modern Trusted Execution Environments (TEEs) that allows remote parties to verify that a host is running binary within a TEE. Remote attestation can be realized through a digital signature scheme. For example, a trusted component on the host computes a digest of the binary, e.g., using a hash function, and signs the digest. The signature is then sent to a remote verifier that checks: (1) that the digest of the binary matches a reference value (e.g., a reciprocally computed hash); and (2) the validity of the received signature by using the verification key corresponding to the signing key of the trusted component.

[0003] Because TEEs are involved in privacy-sensitive applications, remote attestation schemes are designed with anonymity protections for the hosts. In particular, remote attestation schemes use group signatures that allows a signing host to remain anonymous within a group of authorized signers (hosts). In these implementations, a remote verifier learns that a given binary is running within a TEE-enabled host, but it does not learn the particular host where the binary is running. Enhanced Privacy ID (EPID) is a group signature scheme with revocation capabilities that is used for remote attestation in modern TEEs such as Intel SGX.

[0004] Signature schemes are susceptible to subversion attacks, i.e., where a malicious party surreptitiously modifies a cryptographic scheme with the intent of subverting its security. For example, an adversary may subvert the signing algorithm to exfiltrate information to verifying parties. Subversion attacks can be particularly relevant for remote attestation mechanisms of modern TEEs for mainly two reasons. First, the implementation details of the algorithms that constitute the signature scheme may be unavailable for inspection. Second, a subverted signature scheme may exfiltrate the private signing key, or any other identifying information of the host, thereby compromising host anonymity.

[0005] Existing approaches to enhance signature schemes with subversion-resilience rely either on unique signatures or on randomizable signatures. A signature scheme produces unique signatures if, with a fixed verification key, each message has one and one only valid signature. A signature scheme is randomizable if, given a valid signature σ on a message m, any computer can produce a valid signature σ' on m, such that σ' is indistinguishable from a fresh signature on m. For example, Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi, "Subversion-Resilient Signature Schemes," 22nd ACM Conference on Computer and Communications Security, pp. 364-375 2015 (2015) has shown that unique signatures are resilient to signing algorithm subversion, as are randomizable signatures if a source of trusted randomness is available to the party that randomizes the signature.

[0006] J. Camenisch et al.: "Anonymous Attestation with Subverted TPMs", IACR, International Association for Cryptologic Research, vol. 20170628:145434, June 28, 2017, pages 1-79 propose a subversion-resilient direct anonymous attestation (DAA) scheme where the host and the trusted execution environment hold multiplicative shares of a secret signing key. The trusted execution environment, by using its share of the signing key, outputs a partial signature, e.g. St. Next, the host creates the remaining part of the signature, e.g. S_h, by using its share of the signing key. The host also merges the two partial signatures St and S_h in a single one, e.g. S. Finally, the host creates a proof of knowledge of signature S that is used as the group signature and sent to the verifier.

[0007] WO 2017/049111 A1 describes various aspects of group signature schemes with probabilistic revocation. In one example, a computing device can map an alias token to an alias code comprising a plurality of alias code segments.

SUMMARY

[0008] An embodiment of the present invention provides a computer-implemented method that includes receiving, by a host that comprises an intermediate processing system and a trusted execution environment, request for remote attestation of a reference binary from a remote verifier; deploying the reference binary in the trusted execution environment based on the received a request; and calculating, with the trusted execution environment and in response to the remote attestation request, an original digital signature based on the reference binary. The method further includes receiving, by the intermediate processing system, an original message from the trusted execution environment. The original message includes an original digital signature authored by the trusted execution environment. The method includes computing, by the intermediate processing system, a proof of knowledge for the original digital signature; modifying, by the intermediate processing system, the original message by replacing the original digital signature with the proof of knowledge; and forwarding, by the intermediate processing system, the modified original message comprising the proof of knowledge to the remote verifier.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] Embodiments of the present invention will be described in even greater detail below based on the exemplary figures. The invention is not limited to the exemplary embodiments. All features described and/or illustrated herein can be used alone or combined in different combinations in embodiments of the invention. The features and advantages of various embodiments of the present invention will become apparent by reading the following detailed description with reference to the attached drawings which illustrate the following:

FIG. 1 is a block diagram of an exemplary authentication system;

FIG. 2 is a block diagram of an exemplary authentication method; and

FIG. 3 is a block diagram of an exemplary processing system.

DETAILED DESCRIPTION

[0010] TEEs, such as Intel SGX, allow remote verifiers to check that a known piece of code is running untampered within a TEE-enabled host. This can be achieved through a trusted system component that certifies (i.e., signs) code running in a local TEE. TEEs can rely on a group signature scheme with revocation capabilities known as Enhanced Privacy ID (EPID). Group signatures appear in remote attestation to anonymize hosts. For example, a remote verifier learns that a piece of code is running within a TEE-enabled host by inspecting a group signature, but the remote verifier does not learn the particular TEE-enabled host running the code.

[0011] As described above, existing TEE signature schemes are not subversion-resistant. That is, if the signing algorithm is subverted, the TEE subsystem may exfiltrate identifying information via the signatures it produces. The present application discloses embodiments for providing subversion-resilience to group signature schemes that are not unique/randomizable (e.g., existing TEE signature schemes), thereby making the remote attestation mechanism of modern TEEs robust to signing algorithm subversion attacks. Therefore, the present invention provides a subversion-resilient remote attestation for Trusted Execution Environments (TEEs).

[0012] Embodiments of the present invention modify TEE signature schemes to make them subversion-resistant. Embodiments of the present invention provide mechanism to enhance EPID with resilience against subversion attacks. In embodiments of the present invention, a host retains the signature produced by the TEE and provides a verifier with a zero-knowledge proof of knowledge of a valid signature as output by the TEE.

[0013] The present invention provides a mechanism to provide subversion resilience to signature schemes used for remote attestation of TEEs that are not unique nor randomizable. Furthermore, the present invention enables a subversion resilient EPID signature scheme. According to an embodiment of the present invention, a host retains a signature generated on a reference binary by the TEE. To offer the verification functionality required for remote attestation, the host uses the signature on the binary provided by the TEE to compute a zero-knowledge proof of knowledge of that signature. Due to the soundness of the proof, a verifier learns that the reference binary is running within a TEE. Due to the zero-knowledge property of the proof, no further information is exfiltrated to the verifier. In an embodiment, the present invention accounts for the revocation mechanisms currently used in EPID. In an embodiment, the present invention modifies the host and the verifier without modifying the existing TEE.

[0014] An embodiment of the present inventions provides a method for subversion resilient attestation for trusted execution environments, which includes the following operations: (1) replacing the EPID signature provided by a TEE during remote attestation with a proof of knowledge of that signature; and (2) randomizing the revocation token of EPID provided by a TEE platform during remote attestation before sending it to a verifier.

[0015] An embodiment of the present invention provides a method of attestation for a trusted execution environment (TEE) that includes: deploying, by a host having thee TEE, a reference binary in the TEE; receiving, by the host, a request for remote attestation of the reference binary from a remote verifier; issuing, by the TEE of the host, a group signature having a revocation token on the reference binary; withholding, by the host, the group signature; creating, by the host, a proof of knowledge of the group signature; randomizing, by the host, the revocation token; and sending, by the host, the proof of knowledge and the randomized revocation token to the remote verifier. In an embodiment, the group signature may be an Enhanced Privacy Identification (EPID) signature.

[0016] An embodiment of the present invention provides a system for subversion resilient attestation for trusted execution environments, the system including a host that is a TEE-enabled platform and a verifier that is remote from the host. The host deploys a reference binary in a TEE subsystem. The remote verifier issues a request for remote attestation of the reference binary. The TEE subsystem on the host issues a (EPID) signature on the reference binary; the signature contains a revocation token that will be used in case this signature will be placed in a revocation list. The host withholds the signature and creates a proof of knowledge of such signature. The host randomizes the revocation token. The host

sends to the verifier the proof of knowledge on the signature and the randomized revocation token. The verifier checks the proof of knowledge and retains the revocation token for revocation, if needed.

[0017] An embodiment of the present invention provides a host that includes a processor and a memory, the memory having processor executable instructions that, when executed by the processor, cause the processor to perform the following operations for attestation for a trusted execution environment (TEE): deploying a reference binary in the TEE; receiving a request for remote attestation of the reference binary from a remote verifier; receiving, from the TEE, a group signature having a revocation token on the reference binary; withholding the group signature; creating a proof of knowledge of the group signature; randomizing, the revocation token; and sending the proof of knowledge and the randomized revocation token to the remote verifier.

[0018] An embodiment of the present invention provides a non-transitory computer readable medium having processor executable instructions that, when executed by a processor, causes the processor to perform the following operations for attestation for a trusted execution environment (TEE): deploying, by a host having a TEE, a reference binary in the TEE; receiving, by the host, a request for remote attestation of the reference binary from a remote verifier; receiving, from the TEE of the host, a group signature having a revocation token on the reference binary; withholding, by the host, the group signature; creating, by the host, a proof of knowledge of the group signature; randomizing, by the host, the revocation token; and sending, by the host, the proof of knowledge and the randomized revocation token to the remote verifier.

[0019] Disclosed is a computer-implemented method including: receiving an original message from a trusted execution environment, the original message including an original digital signature authored by the trusted execution environment; computing a proof of knowledge for the original digital signature; and modifying the original message by replacing the original digital signature with the proof of knowledge.

[0020] In an embodiment, the method includes: receiving a request for remote attestation of a reference binary from a remote verifier; calculating, with the trusted execution environment and in response to the remote attestation request, the original digital signature based on the reference binary. The original digital signature includes an original revocation token, and the modifying of the original message includes replacing the original revocation token with a randomized revocation token. The method further includes transmitting, to the verifier, the modified original message including the proof of knowledge and the randomized revocation token.

[0021] In an embodiment, the original message includes an original revocation token prepared by the trusted execution environment and the method includes: randomizing the original revocation token; modifying the original message by replacing the original revocation token with the randomized revocation token.

[0022] In an embodiment, the original message includes a destination address and the method includes transmitting the modified message to the destination address. In an embodiment, the original revocation token is randomized by modifying one or more fields of the original revocation token based on a randomly selected parameter.

[0023] In an embodiment, the trusted execution environment locally stores a trusted private key, the trusted execution environment authors the original digital signature with the trusted private key, and the randomized revocation token includes parameters sufficient to revoke the trusted private key. In an embodiment, a host processing system includes an intermediate processing system and the trusted execution environment and the intermediate processing system performs the receiving, the computing, and the modifying.

[0024] In an embodiment, the method includes: receiving a request from a verifying processing system for remote attestation of a reference binary; and deploying the reference binary in the trusted execution environment based on the received request.

[0025] In an embodiment, the original message includes a digest computed by the trusted execution environment based on the reference binary. In an embodiment, the digest includes a hash of the reference binary.

[0026] Disclosed is a processing system including one or more processors configured to: receive an original message from a trusted execution environment, the original message including an original digital signature authored by the trusted execution environment; compute a proof of knowledge for the original digital signature; and modify the original message by replacing the original digital signature with the proof of knowledge.

[0027] In an embodiment, the one or more processors are configured to: receive a request for remote attestation of a reference binary from a remote verifier; calculate, within the trusted execution environment and in response to the remote attestation request, the original digital signature based on the reference binary. The original digital signature includes an original revocation token, and the one or more processors are configured to modify the original message by replacing the original revocation token with a randomized revocation token. The one or more processors may be further configured to transmit, to the verifier, the modified original message including the proof of knowledge and the randomized revocation token.

[0028] In an embodiment, the processing system includes an intermediate processing system and the trusted execution environment. At least one of the intermediate processing system and the trusted execution environment include the one or more processors.

[0029] Disclosed is a non-transitory computer-readable medium including code for configuring one or more processors

to: receive an original message from a trusted execution environment, the original message including an original digital signature authored by the trusted execution environment; compute a proof of knowledge for the original digital signature; and modify the original message by replacing the original digital signature with the proof of knowledge.

[0030] In an embodiment, the non-transitory computer-readable medium includes code for configuring the one or more processors to: receive a request for remote attestation of a reference binary from a remote verifier; calculate, within the trusted execution environment and in response to the remote attestation request, the original digital signature based on the reference binary. The original digital signature includes an original revocation token, and the code for modifying the original message includes code for configuring the one or more processors to replace the original revocation token with a randomized revocation token. The code further configures the one or more processors to transmit, to the verifier, the modified original message including the proof of knowledge and the randomized revocation token.

[0031] Referring to FIG. 1, an authentication system 100 can include a host 110 (also called a host processing system), a remote verifier 120 (also called a verifying processing system), and an issuer/authority 130 (also called an issuing processing system). Host 110 (e.g., the host processing system) can include a TEE 112 (also called a secure enclave) and an intermediate processing system 114. Both TEE 112 and the intermediate processing system 114 can be discrete elements of host 110. TEE 112 can be isolated within host 110 such that information TEE 112 transmits must route through intermediate processing system 114. Host 110 can be distributed such that TEE 112 and intermediate processing system 114 are remote.

[0032] TEE 112 can periodically create a message for transmission to verifier 120. TEE 112 can be configured to create such an original message at the request of intermediate processing system 114 and/or verifier 120. The original message can include an original digital signature. TEE 112 can produce the original digital signature with an internally stored (e.g., hard-coded) first private key. The first private key may be one of a first group of private keys cryptographically paired with a first public key.

[0033] In the absence of subversion, verifier 120, upon receiving the original message, would only be able to confirm a private key within the first group authored the original digital signature. The verifier 120 would not be able to specifically identify the first private key as the author. However, as stated above, a TEE 112 may be compromised so as to encode uniquely identifying information within the original message (e.g., within the original digital signature of the original message).

[0034] Therefore, intermediate processing system 114 can be configured to intercept the message and replace the digital signature with a zero-knowledge proof of knowledge ("ZKPK") such as a non-interactive zero knowledge ("NIZK") proof.

[0035] Intermediate processing system 114 can forward the sanitized message to verifier 120. By doing so, intermediate processing system 114 can prevent TEE 112 from leaking (exfiltrating) information that would uniquely identify host 110 through the digital signature. Therefore, intermediate processing system 114 can make TEE 112 subversion-resilient.

[0036] Verifier 120 can analyze the ZKPK to confirm that intermediate processing system 114 received an original message with a valid digital signature from TEE 112. Verifier 120 can perform an action based on such a confirmation. For example, verifier dispatch a secret message (e.g., a private symmetric key, an access token, etc.) to host 110 or a third-party processing system based on the confirmation.

[0037] In addition to the digital signature and underlying substantive content (e.g., a hash of binary present within TEE 112), the original message can include an original revocation token. In the absence of subversion, TEE 112 can randomly (e.g., pseudo-randomly) select the revocation token from a set of possibilities.

[0038] As with the digital signature, intermediate processing system 114 can substitute the original revocation token with a sanitized (i.e., modified) revocation token. Intermediate processing system 114 can do so by re-randomizing the original revocation token (e.g., by selecting another revocation token from the set of possibilities). By doing so, intermediate processing system 114 can prevent TEE 112 from leaking information that would uniquely identify host through the original revocation token.

[0039] The above functionality can be implemented through a signature scheme Σ including the following algorithms (also called protocols): Setup, Join, Sign, HostSetup, HostProve, Verify, RevokeP, and RevokeS.

[0040] Algorithm Σ.Setup can be run by the issuer (e.g., issuer 130 in FIG. 1) to setup the framework for the signature scheme. Σ.Setup can begin with an initialization algorithm: Init(1λ) $\rightarrow$- pub. This algorithm takes as input the security parameter λ and outputs public parameter pub. Σ.Setup can operate on public parameter pub to produce a group public key gpk and an issuer secret key isk: Setup(pub) $\rightarrow$- (gpk, isk). Group public key gpk and issuer secret key isk can form an asymmetric keypair. As shown in FIG. 1, the issuer 130 can be a processing system (called the issuing processing system) separate from both host 110 and verifier 120.

[0041] Σ.Join can be an interactive join protocol that enables a TEE to obtain group membership credentials. In an embodiment, $\text{Join}_{I,H_i,M_i}\langle(\text{gpk, isk}), \text{gpk, gpk})\rangle \rightarrow \langle b, (b, \text{hvt}_i), \text{sk}_i\rangle$, a three-party protocol between the issuer J, a host H_i (e.g., intermediate processing system 114) and a chip $\mathcal{M}_i$ (e.g., TEE 112). The issuer inputs (gpk, isk), while the other parties only input gpk. In an embodiment, at the end of the protocol, *I* obtains a bit b indicating if the protocol terminated

successfully, M_i (*e.g.*, *TEE* 112) obtains private key sk_i, and H_i obtains a host verification token hvt_i and the same bit *b* as *I*. As further discussed below, $\langle d, e, f\rangle \leftarrow P_{A,B,C}\langle a, b, c\rangle$ can be an interactive protocol P between parties *A, B* and C where *a, b,* c (resp. *d, e, f*) are the local inputs (resp. outputs) of *A, B* and C, respectively.

[0042] A TEE 112 equipped with group membership credentials can produce signatures via Σ.Sign (also called Σ.Sig): Sign(gpk, sk_i, bsn, M, SigRL) →- ⊥/(σ, π_σ). According to this embodiment, the signing algorithm takes as input the group public key gpk, a private key sk_i, a basename bsn (e.g., an arbitrary string), a message *M*, and a signature-based revocation list SigRL. The signing algorithm outputs a signature σ and a proof π_σ, or an error ⊥ (if SigRL contains a signature produced with private key sk_i).

[0043] Σ.HostSetup can be performed once per host. In an embodiment, this algorithm sets up (e.g., configures intermediate processing system 114 to perform) a non-interactive zero knowledge ("NIZK") proof system for the relation R_{EPID}. R_{EPID} can contain all tuples of the following elements: a group public key gpk, a list of revoked signatures SigRL, a basename bsn, a message M, and a randomized revocation token (B', K'). Witnesses can be valid signatures out by Sign(gpk, ski, bsn, M, SigRL) using a membership key ski and a random element α.

[0044] In an embodiment detail, a tuple ((gpk, SigRL, bsn, M, (B', K')), (σ, α))$\in R_{EPID}$ if and only if:

1. SigRL can be parsed as $\{B_i,K_i\}_{1\le i\le|SigRL|}$.

2. $\sigma=(\sigma_0, \sigma_1,...,\sigma_{|SigRL|})$

3. σ_0=(B, K, T, c, s_x, s_f, s_a, s_b) where:

   a. c=H(gpk, B, K, T, R_1', R_2', m)
   b. $R_1'=B^{sf}K^{-c}$
   c. $R_2'=e(T,g_2)^{-}$

$$^{sx}e(h_1,g_2)^{sf}e(h_2,g_2)^{sb}e(h_2,w)^{sa}(e(g_1,g_2)/e(T,w))^{c}$$

4. σ_i is a valid zero-knowledge proof for SPK{(f}: K=B^f Λ K_i<>B_i^f}(m), for 1≤i≤|SigRL|. SPK can be a signature of knowledge (e.g., a proof of knowledge where the hash used as a challenge is a function of message M).

5. $B'=B^\alpha$

6. $K'=K^\alpha$

[0045] By performing items 1-4, intermediate processing system 114 can ensure that the original signature is valid and by performing items 5-6, intermediate processing system 114 can ensure that the sanitized pair (B', K') is a valid re-randomization of a revocation token. In an embodiment, the zero-knowledge proof is a function of α. For example, a witness for R_{EPID} can be a pair σ, α

[0046] During Σ.HostSetup, intermediate processing system 114 can be configured to perform Init(1^λ)→ω, where λ is the above-discussed security parameter applied by the issuer during Σ.Setup, then set gpk*=(ω, gpk), where gpk is the group public key generated during Σ.Setup. Intermediate processing system 114 can send gpk* to the verifier 120. Intermediate processing system 114 may perform the algorithm at every interaction with a verifier (i.e., for every message sent to a verifier) in case the host 110 requires any pairs of interactions with a given verifier to be unlinkable.

[0047] Intermediate processing system 114 can perform algorithm Σ.HostProve(σ)→π (also called HostSanitize) to compute the proof of knowledge (e.g., NIZK) of a signature. Given a signature σ on a message M as authored by TEE 112, intermediate processing system 114 can parse σ as (σ_0,σ_i,...,σ_n) and σ_0 as (B,K,T,c,s_x,s_f,s_a,s_b). Intermediate processing system 114 can perform the following:

1. Compute (B', K')=(B^α, K^α) for a random α in Z_p

2. Run Prove(ω, (gpk, SigRL, m, (B', K')), σ)→π

3. Sends α^*=((B', K'), π)

[0048] Algorithm Σ.Verify(gpk*,m, α^*,SigRL,Priv-RL)→0/1 can be performed by verifier 120. In an embodiment, the algorithm parses α^* as ((B', K'), π) and gpk* as (co, gpk). If ProofVerify(ω, (gpk, m, (B',K'), π)→1 and K<>B^{fi} for each

$f_i \in$ Priv-RL, then the algorithm outputs 1; otherwise the algorithm outputs 0. In an embodiment, the verifier applies α^*. The notation α^* does not imply a relationship between α^* and α, although in an embodiment, α^* proves knowledge of α.

[0049] Issuer 130 can perform algorithm Σ.RevokeP(gpk, Priv-RL, sk_i)$\rightarrow$Priv-RL'. Issuer 130 can take revocation list Priv-RL and secret key ski=fi, and outputs Priv-RL'=Priv-RL U {fi}

[0050] Issuer 130 can perform algorithm Σ.RevokeS(gpk, Priv-RL, SigRL, m, σ)$\rightarrow$SigRL'/$\perp$. Given revocation list SigRL and a signature σ^*, if Σ.Verify(gpk*,m, α^*,SigRL,Priv-RL)$\rightarrow$1, then output SigRL'=SigRL U {(B',K'}; otherwise output 1.

[0051] FIG. 2 illustrates an exemplary remote attestation method. At block 202, a verifying processing system 120 can demand remote attestation of software (e.g., reference binary) stored on a host processing system 110 as a condition precedent to performing an activity (e.g., issuing a symmetric key or other valuable data to host processing system 110) Host processing system 110 can include TEE 112 and intermediate processing system ("IPS") 114.

[0052] At block 204, intermediate processing system 114 can receive the demand, and in response, forward the subject of the demand (e.g., software such as reference binary present within intermediate processing system 114) to TEE 112 for computation of a digest authenticating the software. The digest can be, for example, a hash of the software computed according to a hashing algorithm supplied by the verifying processing system 120.

[0053] At block 206, TEE 112 can compute the digest. At block 208, TEE 112 can create a message addressed to verifying processing system 120. The message can include an original digital signature authored by TEE 112 and an original revocation token. TEE 112 can compute the digital signature based on group public key gpk, secret key sk_i, basename bsn, substantive content of the message M (e.g., the digest), and the signature-based revocation list SigRL. TEE 112 can transmit the message to verifying processing system 120.

[0054] At block 210, intermediate processing system 114 can intercept the message. At block 212, intermediate processing system 114 can compute a sanitized digital signature and a sanitized revocation token. Intermediate processing system 114 can compute the sanitized digital signature based on the group public key gpk, the basename bsn, the substantive message content M (e.g., the digest), the original digital signature, the signature-based revocation list SigRL, and the host verification token hvt_i. Intermediate processing system 114 can compute the sanitized revocation token based on the original revocation token and a randomly selected parameter. In an embodiment, intermediate processing system 114 can raise each parameter of the original revocation token (e.g., B and K) to the power of the randomly selected parameter.

[0055] At block 212, intermediate processing system 114 can replace the original digital signature and the original revocation token with the sanitized digital signature and the sanitized revocation token. The original digital signature and original revocation token can be stripped from the message. At block 214, intermediate processing system 114 can transmit the sanitized (also called modified) message to verifying processing system 120. The sanitized message can include the substantive message content M (e.g., the digest), the sanitized digital signature, and the sanitized revocation token. The substantive message content M can be omitted in scenarios where it is already available at the verifier.

[0056] At block 216, verifying processing system 120 can confirm that the sanitized digital signature includes a proof of knowledge of the original digital signature and that the substantive message content M (e.g., the digest) is an expected value (e.g., by computing a reciprocal hash of the reference binary). At block 218, and based on confirming that the sanitized digital signature and the substantive message content are appropriate, verifying processing system 120 can supply information to intermediate processing system 114 (e.g., a symmetric key, a digital signature, etc.).

[0057] The following description is of proof of relations over discrete logarithms and signature of knowledge according to an embodiment.

[0058] A proof of knowledge of a discrete logarithm of an element y of a multiplicative group G with respect to a base g is denoted as PK{(x) : $y = g^x$}. See, e.g., Claus Schnorr, "Efficient Identification and Signatures for Smart Cards," Journal of Cryptology 4(3) (1991)). This can be accomplished as follows. The prover picks random r and sends $t=g^r$. The verifier responds with random challenge c. The prover sends s=r+cx. The verifier accepts the proof if $g^s=ty^c$.

[0059] A proof of knowledge of a representation of an element $y \in G$ with respect to several bases $(g_1,...,g_v) \in G^v$ can be denoted PK{$(x_1, ... , x_v)$: $y = g^{x1}g^{x2}...g^{xv-1}g^{xv}$}. See, e.g., David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf, "An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations," EUROCRYPT (1987) . The prover picks random $r_l,...,r_v$ and sends $t_l,...t_v$ where $t_i=g^{ri}$. The verifier responds with random challenge c. The prover sends $s_l,...,s_v$ where $s_i=r_i+cx_i$. The verifier accepts the proof if $\Pi_{1\leq i\leq v} g_i^{si} = \Pi_{1\leq i\leq v} t_i y^c$.

[0060] A proof of equality of discrete logarithms of two group elements $y_1,y_2 \in G$ to bases $g_1,g_2 \in G$, can be denoted PK{(x) : $y_1=g_1^x \Lambda y_2=g_2^x$}. See, e.g., David Chaum and Torben Pedersen, "Wallet Databases with Observers," CRYPTO (1).

[0061] The prover picks random r, computes $t_1=g_1^s$, $t_2=g_2^s$, and sends (t_1, t_2). The verifier responds with random challenge c. The prover sends s=r+cx. The verifier accepts the proof if $g^s=t_1y_1^c$ and $g^s=t_2y_2^c$.

[0062] A proof of inequality of discrete logarithms of two group elements $y_1,y_2 \in G$ to bases $g_1,g_2 \in G$, is denoted PK{(x) : $y_1=g_1^x \Lambda y_2<>g_2^x$}. See, e.g., Jan Camenisch and Victor Shoup, "Practical Verifiable Encryption and Decryption of Discrete Logarithms," CRYPTO (2003)). The prover picks random r and sends $c=(g_2/y_2)^r$. Prover and verifier execute the protocol PK{(a, b): $C=g_2^a(1/y_2)^b \Lambda 1 = g_1^a(1/y_1)^b$}. The verifier accepts if it accepts the previous proof and if C<>1.

[0063] The above interactive proofs may be converted into non-interactive proofs by replacing the verifier's challenge with a hash computed over the protocol transcript up to the point where the challenge must be provided. For example, in the proof of knowledge of a discrete logarithm of an element $y \in G$ with respect to a base g, the challenge can be computed as c=H(g, y, t). Further, the proof can be converted in a signature of knowledge on a message m, if the input to the hash function includes m. The notation SPK{(a): y=ga}(m) is used to denote a signature on a message m obtained in this way.

[0064] The following description is of Non-Interactive Zero-Knowledge Proof of Knowledge according to an embodiment.

[0065] A non-interactive zero-knowledge (NIZK) proof system for a relation R can be a tuple (Init, Prove, VerifyProof) of probabilistic polynomial-time algorithms such that: Init on input a security parameter outputs a common reference string ω; Prove(ω, x, w) on input (x, w)$\in$R, outputs a proof π; VerifyProof(ω, x, π) given an instance x and a proof π, outputs 0 (reject) or 1 (accept). A NIZK is sound if a prover has negligible chances of producing convincing proofs for statements outside of R, while the NIZK is zero-knowledge if a verifier learns nothing from a proof (apart from the fact that the proof is valid). NIZK for arbitrary relations are available, for example Jens Groth and Amit Sahai, "Efficient Non-interactive Proof Systems for Bilinear Groups," IACR Cryptology ePrint Archive 155 (2007).

[0066] The following description is of a BBS+ Signature scheme according to an embodiment:

[0067] An EPID can be built on top of a BBS+ signature scheme. See, e.g., Man Ho Au, Willy Susilo, and Yi Mu, "Constant-Size Dynamic k-TAA," SCN (2006) (discussing BBS+ signatures).

[0068] Here, (G_1, G_2) are a suitable bilinear group pair of some prime order p. And, $e:G_1 x G_2 \rightarrow G_T$ are a computable bilinear pairing function. The BBS+ scheme can unfold as follows:

[0069] BBS.KeyGen(1^λ)$\rightarrow$(pk, sk). The key generation algorithm takes as input security parameter λ. It randomly picks (g_1, h_1, h_2)$\in {G_1}^3$, $g_2 \in G_2$, and $\gamma \in Z_q$. It sets $w=g_2^{\gamma}$ and outputs pk=(p, G_1, G_2, G_T, g_1, h_1, h_2, w) and sk=y.

[0070] BBS.Sign(pk, sk, m)$\rightarrow\sigma$. The signing algorithm takes a pair of public-private keys and a message m. It outputs a signature σ=(A, x, y), where (x, y) are random elements of Z_p^* and $A=(g_1 {h_1}^m {h_2}^y)^{1/(x+\gamma)}$.

[0071] BBS.Verify(pk, m, σ)$\rightarrow$0/1. The verification algorithm outputs 1 (i.e., valid signature) if $e(A, {g_2}^x w)=e(g_1 {h_1}^m {h_2}^y, g_2)$; otherwise it outputs 0.

[0072] The following description is of Enhanced Privacy ID (EPID) according to an embodiment:

[0073] Enhanced Privacy ID (EPID) is a group signature scheme with provisions for revocation. EPID includes four different types of entities: (1) an issuer I who manages group membership; (2) a revocation manager R who manages revocation of group members (As shown in FIG. 1, I and R can be the same entity); (3) a set of group members; and (4) a set of verifiers.

[0074] Group membership is regulated by I, who provides a member Mi with a secret key ski via a join protocol. Member Mi uses its secret key to issue signatures that anybody can verify. A valid signature convinces the verifier that the signer is a member of the group, but does not allow the verifier to learn Mi's identity.

[0075] A revocation manager R can, if needed, revoke group membership. In particular, EPID provides two different kind of revocation lists: one is called Priv-RL and is constituted by the secret keys of the revoked members, while the other is called SigRL and includes (part of the) signatures generated by revoked members. The latter is used when a member must be revoked, but its private key is not available.

Notation:

[0076] In the following, <c, d>$\leftarrow {P_{A,B}}$<a, b> is referred to as an interactive protocol P between parties A and B, where A's private input is a and B's private input is b. The protocol returns private output c to A and private output d to B.

[0077] EPID can be described as a tuple (Setup, Join, Sign, Verify, RevokeP, RevokeS) defined as follows:

[0078] Setup(1^λ)$\rightarrow$(gpk, isk). This algorithm is run by the issuer I. It takes a security parameter λ as input and outputs a group public key gpk and an issuer secret key isk.

[0079] <1, sk_i>$\leftarrow {Join_{I,Mi}}$<(gpk, isk), gpk>. This is an interactive protocol between issuer I and prospective member Mi. Both parties input the group public key, while the issuer also inputs its secret key. At the end of the protocol the issuer outputs nothing (1) while member Mi outputs secret key ski.

[0080] Sign(gpk, ski, m, SigRL)$\rightarrow\perp/\sigma$. The signing algorithm takes as input the group key public gpk, signing key ski, a message m, and a signature revocation list SigRL. It outputs either a valid signature σ or an error message 1.

[0081] Verify(gpk, Priv-RL, SigRL, m, σ)$\rightarrow$0/1. The signature verification algorithm takes as input the group public key gpk, both revocation lists Priv-RL and SigRL, a message m, and a signature a. It outputs 1 if σ is a valid signature on m; otherwise it outputs 0.

[0082] RevokeP(gpk, Priv-RL, sk_i)$\rightarrow$Priv-RL'. This algorithm outputs an updated revocation list by adding secret key ski to the input list Priv-RL.

[0083] RevokeS(gpk, Priv-RL, SigRL, m, σ)$\rightarrow$SigRL'/$\perp$. If signature σ is valid for message m, this algorithm outputs an updated revocation list by adding a part of σ to the input list Priv-RL; otherwise the algorithm outputs an error message 1.

[0084] The following description is of an instantiation of Enhanced Privacy ID (EPID) according to an embodiment.

[0085] An example instantiation of EPID-adopted by Intel SGX-is discussed in Ernie Brickell and Jiangtao Li, "Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation," SocialCom/PASSAT (2010) ("Brickell") NIZK-e.g., as proposed in Brickell-may be used by an embodiment of the present invention. At a high level, a dedicated service by the platform manufacturer acts as issuer and revocation manager. There may be several groups. However, without loss of generality, the present example assumes that there is only one group.

[0086] The issuer sets up the group via the Setup algorithm. Any platform can become a member of the group by choosing a secret key f, and engaging with the issuer I in the Join interactive protocol. During the protocol, the platform outputs a BBS+ signature σ=(A,x,y) over f, computed with the issuer secret key. The signature σ and the secret key f constitute the platform's credentials as a group member.

[0087] A signature by a TEE is essentially a zero-knowledge proof that the TEE has a BBS+ signature on its secret key f. Further, given a revocation list SigRL of signatures produced by revoked members, a valid signature must also include a proof that the secret key f may have not produced any of the signatures in SigRL. Finally, the TEE also picks a random base B, computes $K=B^f$ and proves in zero-knowledge that K was computed correctly from B and f. The pair (B, K) is denoted as a revocation token; this is provided to the verifier and may be added to SigRL by the revocation manager R later on.

[0088] The following is an algorithm implemented by embodiments of the present invention:

[0089] $Setupf(1^{\lambda})\rightarrow(gpk, isk)$. On input security parameter λ, it runs $BBS.KeyGen(1^{\lambda})\rightarrow(pk, sk)$ and sets gkp=pk and isk=sk.

[0090] $<\perp,sk_i>\leftarrow Join_{I,Mi}<(gpk, isk), gpk>$. This is an interactive protocol between platform Mi and issuer I that unfolds as follows:

1. M_i picks $(f, y') \in Z_p^2$ and computes $t=h_1^f h_2^{y'}$ and produces a proof $\pi=PK\{(f, y): t=h_1^f h_2^{y'}\}$.

2. The issuer verifies π, computes $A=(g_1 h_1^m h_2^{y''})^{1/(x+\gamma'')}$ where (x, y") are random elements of Z_p, and outputs (A, x, y)

3. Mi computes y=y'+y" and checks that $e(A, g_2^x w)=e(g_1 h_1^m h_2^y, g_2)$. It outputs ski=(A, x, y, f).

[0091] $Sign(gpk, ski, m, SigRL)\rightarrow\perp/\sigma$. Given the group public key gpk, a member's credentials ski=(A, x, y, f) and a message $m\in\{0,1\}^*$, the signing algorithm unfolds as follows:

1. Pick random $B\in G_T$ and random $a\in Z_p$ and compute $K=B^f$, $b=y+ax$, $T=Ah_2^a$

2. Compute $SPK\{(x, f, a, b): B^f=K \wedge e(T,g_2)^{-x}e(h_1,g_2)^f e(h_2,g_2)^b e(h_2,w)^a=e(T,w)/e(g_1,g_2)\}(m)$

   a. Compute $R_1=B^{rf}$, $R_2=e(T, g_2)^{-rx}e(h_1,g_2)^{rf}e(h_2, g_2)^{rb}e(h_2, w)^{ra}$, where r_x, r_f, r_a, r_b are random elements of Z_p.
   b. Compute $c=H(gpk, B, K, T, R_1, R_2, m)$ and $s_x=r_x+cx$, $s_f=r_f+cf$, $s_a=r_a+ca$, $s_b=r_b+cb$.

3. Set $\sigma_0=(B, K, T, c, s_x, s_f, s_a, s_b)$.

4. Given $SigRL=\{B_i,K_i\}_{1\leq i\leq|SigRL|}$, compute $\sigma_i=SPK\{(f\}: K=B^f \wedge K_i<>B_i^f\}(m)$ for $1\leq i\leq|SigRL|$.

5. If any of the zero-knowledge proofs above fails, output $\perp$, otherwise output $\sigma=(\sigma_0, \sigma_1,..., \sigma_{|SigRL|})$.

[0092] $Verify(gpk, Priv\text{-}RL, SigRL, m, \sigma)\rightarrow 0/1$. Let $\sigma=(\sigma 0, \sigma 1,..., \sigma|SigRL|)$ where $\sigma 0=(B, K, T, c, sx, sf, sa, sb)$.

1. Verify that $(B, K, T, s_x, s_f, s_a, s_b)\in G_T x G_T x G_1 x Z_p^4$.

2. Compute $R_1'=B^{sf}K^{-c}$ and $R_2'=e(T, g_2)^{-sx}e(h_1, g_2)^{sf}e(h_2, g_2)^{sb}e(h_2, w)^{sa}(e(g_1, g_2)/e(T, w))^c$

3. Verify that $c=H(gpk, B, K, T, R_1', R_2', m)$.

4. Given $SigRL=\{B_i,K_i=B_i^{fi}\}_{1\leq i\leq|SigRL|}$, verify that σ_i is a valid zero-knowledge proof for $SPK\{(f\}: K=B^f \wedge K_i<>B_i^f\}(m)$, for $1\leq i\leq|SigRL|$.

5. For each $f_i\in$Priv-RL, check that $K<>B^{fi}$.

[0093] $RevokeP(gpk, Priv\text{-}RL, sk_i)\rightarrow$Priv-RL'. Output Priv-RL'=Priv-RL U {ski}.

[0094] RevokeS(gpk, Priv-RL, SigRL, m, σ)$\rightarrow$SigRL'/$\perp$. If SsGx.Verify(gpk, Priv-RL, SigRL, m, σ)$\rightarrow$1, then output SigRL'=SigRL U {(B',K')}; otherwise output 1.

[0095] Referring to FIG. 3, a processing system 300 can include one or more processors 302, memory 304, one or more input/output devices 306, one or more sensors 308, one or more user interfaces 310, and one or more actuators 312. Processing system 300 can be representative of each of host 110, TEE 112, intermediate processing system 113, verifier 120, and the issuer 130.

[0096] Processors 302 can include one or more distinct processors, each having one or more cores. Each of the distinct processors can have the same or different structure. Processors 302 can include one or more central processing units (CPUs), one or more graphics processing units (GPUs), circuitry (e.g., application specific integrated circuits (ASICs)), digital signal processors (DSPs), and the like. Processors 302 can be mounted on a common substrate or to different substrates.

[0097] Processors 302 are configured to perform a certain function, method, or operation at least when one of the one or more of the distinct processors is capable of executing code (e.g., interpreting scripts), stored on memory 304 embodying the function, method, or operation. Processors 302, and thus processing system 300, can be configured to perform, automatically, any and all functions, methods, and operations disclosed herein.

[0098] For example, when the present disclosure states that processing system 300 performs/can perform task "X" (or that task "X" is performed), such a statement should be understood to disclose that processing system 300 can be configured to perform task "X". Processing system 300 are configured to perform a function, method, or operation at least when processors 302 are configured to do the same.

[0099] Memory 304 can include volatile memory, non-volatile memory, and any other medium capable of storing data. Each of the volatile memory, non-volatile memory, and any other type of memory can include multiple different memory devices, located at multiple distinct locations and each having a different structure. Memory 304 can include cloud storage.

[0100] Examples of memory 304 include a non-transitory computer-readable media such as RAM, ROM, flash memory, EEPROM, any kind of optical storage disk such as a DVD, a Blu-Ray® disc, magnetic storage, holographic storage, an HDD, an SSD, any medium that can be used to store program code in the form of instructions or data structures, and the like. Any and all of the methods, functions, and operations described in the present application can be fully embodied in the form of tangible and/or non-transitory machine-readable code (e.g., scripts) saved in memory 304.

[0101] Input-output devices 306 can include any component for trafficking data such as ports, antennas (i.e., transceivers), printed conductive paths, and the like. Input-output devices 306 can enable wired communication via USB®, DisplayPort®, HDMI®, Ethernet, and the like. Input-output devices 306 can enable electronic, optical, magnetic, and holographic, communication with suitable memory 306. Input-output devices 306 can enable wireless communication via WiFi®, Bluetooth®, cellular (e.g., LTE®, CDMA®, GSM®, WiMax®, NFC®), GPS, and the like. Input-output devices 306 can include wired and/or wireless communication pathways.

[0102] Sensors 308 can capture physical measurements of environment and report the same to processors 302. User interface 310 can include displays, physical buttons, speakers, microphones, keyboards, and the like. Actuators 312 can enable processors 302 to control mechanical forces.

[0103] Processing system 300 can be distributed. Processing system 300 can have a modular design where certain features have a plurality of the aspects shown in FIG. 3. For example, I/O modules can include volatile memory and one or more processors.

[0104] The following disclosure provides further description of exemplary embodiments of the invention.

[0105] Enhanced Privacy ID (EPID) is signature scheme underlying anonymous attestation in Intel SGX. Disclosed is a *subversion resilient* EPID scheme (or SR-EPID). SR-EPID provides the same functionality and security guarantees of the original EPID, despite a potentially subverted chip (e.g., TEE 112). In this design, the host (e.g., intermediate processing system 114 of host 110) acts as a "sanitizer" and ensures no covert channel between the chip (e.g., TEE 112) and the outside world both during enrollment and during attestation (i.e., when signatures are produced). It can do so by leveraging structure preserving signatures in combination with Groth-Sahai proof system. As used herein, the term host 110 can refer to both intermediate processing system 114 and TEE 112 or intermediate processing system 114 specifically.

[0106] The above-described approach has a number of advantages. First, the host (e.g., intermediate processing system 114) bears no secret information-hence, a memory leak does not erode security. Further, the role of sanitizer may be distributed in a cascade fashion among several parties (e.g., multiple components of intermediate processing system 114) so that sanitization becomes effective as long as one of the parties has access to a good source or randomness. This approach is suited for corporate environments where security gateways process outgoing messages before they are released. The signing protocol can be non-interactive, thereby minimizing latency during signature generation. The instantiation of SR-EPID is secure against adaptive corruptions.

[0107] Different from existing approaches, signatures in SR-EPID can be produced by the chip via a non-interactive algorithm. The host can be leveraged to ensure security despite a subverted chip, but is not required to store a secret outside of TEE 112. In SR-EPID the host can act as a sanitizer: it runs a special verification of the signature produced

by the chip; if this verification passes, the hosts strips off a piece of the signature, and forwards a re-randomization of the remaining part to the verifier (e.g., verifier 120).

[0108] The disclosed approach comes with multiple benefits. First, signature generation can be non-interactive and the communication flow can be unidirectional (from the chip to the host, on to the verifier)- just like the original EPID scheme and its instantiation within Intel SGX. This decreases latency and provides more flexibility in the protocol flow as the sanitization of a signature does not need to be done online. Non-interactive signature generation places a higher toll on the signer, but this is not an issue for secure hardware like Intel SGX that leverages the full power of the main processor.

[0109] Another benefit is that the host (i.e., intermediate processing system 114) holds no secret and only needs to store a verification key, which can even be made public without invalidating anonymity or unforgeability. This means that if a memory leak occurs on the host, one has nothing to recover but public information that is useless even if the chip is subverted. This property does not hold in existing approaches, where an attacker may ship a subverted hardware with a hardcoded signing key share, and a memory leak on the host would allow to get the host's share of the signing key; once the full signing key is known, unforgeability and anonymity are essentially lost.

[0110] In an embodiment, the host needs to have access to good randomness that should not be leaked; this is to re-randomize the signatures produced by a possibly subverted chip. However, the re-randomization is non-interactive and requires no long-term secret. Therefore, re-randomization can be executed by multiple parties in a cascade fashion in such a way that the signature is properly re-randomized as long as at least one of these parties has good randomness.

[0111] This *modus operandi* may fit corporate environments where hosts themselves may not be trusted with preventing information exfiltration, and security checks are carried out by one or more company gateways. It is not clear how to achieve such "fault tolerance" in various existing split-signature approaches. One may split the signing key across several parties, but the non-interactive nature of the signing protocol would lead to high latency for signature generation.

[0112] An embodiment formalizes the notion of Subversion-Resilient EPID schemes and includes a construction based on bilinear pairings. The construction leverages structure preserving signatures in combination with Groth-Sahai proof system and achieves security against adaptive corruptions.

[0113] In an embodiment consistent with FIG. 1, the issuer authority *I* 130 (e.g., Intel in case of SGX) manages group(s) and certifies secret keys. A group member is a platform *I* (labeled host 110 in FIG. 1) including a host *H* (*labeled intermediate processing system* 114 *in FIG.* 1), and a chip *M* (*labeled TEE* 112 *in FIG.* 1) holding the signing key. In case of SGX, the chip *M* is the so-called *quoting enclave,* whereas the host *H* is constituted by the remaining hardware and software on the platform.

[0114] The quoting enclave can be the only component within host H with access to the attestation (i.e., EPID) signing key. This certified secret key never leaves the chip as it features tamper-resistant storage. Platform authentication by remote verifiers V is achieved by having the platform issuing a signature on a (challenge) message. By verifying the validity of the signature, the verifier is assured that the platform is a valid group member but cannot identify the platform among those belonging to the same group. Arrows in FIG. 1 depict available communication channels; all communication between the chip and the outside world is mediated by the host (i.e., intermediate processing system 114). The same model holds for anonymous attestation using TPMs; in this case, the chip is the TPM and the host is any other hardware and the software on the same platform.

[0115] In order to guarantee security in the presence of a subverted chip *M*, host to "sanitizes" signatures produced by the chip so to eliminate any covert channel between the chip and the outside world.

[0116] A covert channel could, for example, leverage nonces chosen by the chip during signature generation. Alternatively, a subverted chip may run the join protocol-when the private key is certified by the group manager-with a private key known to the adversary; later on, the adversary may simply use this known private key to break platform anonymity (since, by definition, private key based revocation allows a verifier to tell if a signature has been produced with a given private key). Another option is for the chip to behave honestly during the join protocol, but later use a preloaded private key to produce signatures (since platform anonymity must hold also against a malicious group authority, it is possible for the hardware to be preloaded with one or more certified private keys). The adversary may use that known (preloaded) key and a signature to break platform anonymity.

[0117] To deal with these issues, an embodiment of SR-EPID is designed so that (i) the host (e.g., intermediate processing system 114) participates to the join protocol contributing to the private key of the chip, (ii) each signature output by the chip (e.g., TEE 112) carries a proof (for the host to verify) that the private key used for signing is the very same one certified during the join protocol, and (iii) the host sanitizes signatures to avoid covert channel based on maliciously-sampled nonces.

[0118] $\langle d, e, f\rangle \leftarrow P_{A,B,C}\langle a, b, c\rangle$ is an interactive protocol P between parties *A, B* and C where *a, b,* c (resp. *d, e, f*) are the local inputs (resp. outputs) of *A, B* and C, respectively. As discussed below, an embodiment of SR-EPID can include algorithms (also called protocols) Join, Init, Setup, Sig, Ver, and Sanitize. All the algorithms except Init take one or more public parameters generated by Init as an input. For convenience, this input is implicit in the following description.

[0119] Init(1^{λ}) →- pub. This algorithm takes as input the security parameter λ and outputs public parameters pub.

[0120] Setup(pub) →- (gpk, isk). This algorithm takes the public parameters pub and outputs a group public key gpk and an issuing secret key isk for the issuer I.

[0121] Join_{I,H_i,M_i} ((gpk, isk), gpk, gpk)) →- *(b,* (*b*, hvt_i), sk_i⟩. This is a three-party protocol between the issuer J, a host *Hi* and a chip *Mi.* The issuer inputs (gpk, isk), while the other parties only input gpk. At the end of the protocol, *I* obtains a bit b indicating if the protocol terminated successfully, *Mi* obtains private key sk_i, and *Hi* obtains a host verification token hvt*i* and the same bit b of J.

[0122] Sig(gpk, ski, bsn, M, SigRL) →⊥/(σ,πσ). The signing algorithms takes as input the group public key gpk, a private key sk_i, a basename bsn, a message *M*, and a signature based revocation list SigRL. It outputs a signature σ and a proof π_σ, or an error ⊥ (if SigRL contains a signature produced with sk_i).

[0123] Ver(gpk, bsn, M, σ, SigRL, PrivRL) →- 0/1. The verification algorithm takes in input the group public key gpk, a basename bsn, a message M, a signature σ, a signature based revocation list SigRL, and a private key based revocation list PrivRL. It outputs 0 or 1 if σ is respectively an invalid or a valid signature on M.

[0124] Sanitize(gpk, bsn, M, (σ, πσ), SigRL, hvt*i*) →- ⊥/σ'. The sanitization algorithm takes as input the group public key gpk, a basename bsn, a message *M*, a signature σ with corresponding proof $\pi\sigma$, a signature based revocation list SigRL, and a host verification token hvt*i*. It outputs either ⊥ or a sanitized signature σ'.

[0125] Link(gpk, bsn, M_1, σ_1, SigRL_1, M2, σ_2, SigRL_2) →- 0/1. The linking algorithm takes as input the group public key gpk, a basename bsn, and two message-signature-SigRL triples M_1, σ_1, SigRL_1 and M2, σ_2, SigRL_2. It outputs 1 if both signatures are valid and were created, on the same basename, by the same platform; it outputs 0 otherwise.

[0126] In an embodiment, PrivRL is a set of private keys {sk*i*}*i*, and SigRL is a set of triples {(bsni, *Mi, σi*)} *i*, each including (e.g., consisting of) a basename, a message and a signature.

[0127] *Correctness:* An SR-EPID scheme satisfies correctness if any signature produced by a non-revoked platform passes the verification procedure. More formally, for all pub ← Init(1^λ), all (gpk, gsk) ← Setup(pub), all *(b, (b,* hvt), sk) ← Join((gpk, gsk), gpk, gpk)) such that b = 1, and for any basename bsn, message M, private-key revocation list PrivRL and signature revocation list SigRL, and any signature σ ← Sanitize(gpk, bsn, M, Sig(gpk, sk, bsn, M, SigRL), SigRL, hvt) the following expression is obtained where Σ is the set of signatures produced with sk:

$$\text{Ver}(\text{gpk}, \text{bsn}, M, \sigma, \text{SigRL}, \text{PrivRL}) = 1 \Rightarrow (\text{sk} \notin \text{PrivRL}) \wedge (\Sigma \cap \text{SigRL} = \emptyset)$$

[0128] An exemplary construction includes the following template: (I) The issuer *I* keeps a secret key isk of a (structure-preserving) signature scheme and, in an embodiment, the secret key of a platform is a signature σ_{sp} on a Pedersen commitment [*t*]1 whose opening y is known to the platform only (in other words σ_{sp} is a blind signature on y). (II) The chip generates a signature on a message *M* by creating a signature of knowledge for *M* of a signature σ_{sp} made by *I* on a commitment [*t*]1 that opens to y. To glue the message *M* with σ_{sp} the proof is created using a labeled NIZK, where the label is, indeed, the message M.

[0129] A feature for preventing subversion attacks is to let the host *H* re-randomize every signature produced by the chip *M*, eliminating in this way the possibility that the randomness chosen by *M* in the signature of knowledge be a covert channel. Technically, this is achieved by using re-randomizable NIZKs.

[0130] This feature, however, can be insufficient to counter other possible subversion attacks. Not only the signatures, but also the execution of the Join protocol can give rise to a covert channel between the chip and the adversary. For example, the chip *M* may choose a hardcoded secret y as its secret platform key. For this, the host *H* also contributes to this key. And more in general potential covert channels in the Join protocol can be eliminated by letting the host re-randomize any message and NIZK that go from the chip to the issuer.

[0131] Another potential attack vector is that after the Join protocol is over, a subverted chip may use a completely different secret key y' to generate signatures. In particular a chip subverted by a malicious issuer may come with such y 'together with a valid signature σ'_{sp} for it. To contrast this class of attacks, *M* can be required to output, for each signature σ, a proof πσ that it is generated with the same secret key obtained Join. *H* can check πσ using a dedicated verification token (also obtained at the end of the Join protocol); if the check passes, *H* strips off πσ , and returns a re-randomization of σ.

[0132] Disclosed are another set of techniques to reconcile an extensive use of (Groth-Sahai) NIZKs with the goal of obtaining an efficient SR-EPID scheme.

[0133] An embodiment requires two seemly conflicting properties of NIZK proof systems. On the one hand, the embodiment extracts from the malicious platforms the secret keys after the join protocols are executed; on the other hand, the embodiment uses zero-knowledge and strong derivation privacy to disable any covert channel from an honest platform with a subverted chip. The embodiment can rely on a simulation-extractable (and malleable) NIZK scheme. In an embodiment, any re-randomization algorithm is able to change the hash value to a fresh one without the knowledge of the witness, which is in contrast with the special soundness of the sigma-protocols.

[0134] In a construction, the expensive use of simulation-extractable (and malleable) GS proofs is avoided by using a novel combination of (plain) GS proofs with the random oracle model. At the beginning of the join protocol, the issuer chooses a fresh identifier *id* which the parties hash with a random oracle J, and its output is used as a fresh common-reference string. All the NIZKs sent by the platform during Join will use this fresh common reference string. In this way, by programming the random oracle, in the reduction one can adaptively choose the flavor of common reference string needed, depending on the kind of corruption of the platform.

[0135] However, this strategy may only partially succeed when using GS proof systems. The random oracle can be used to program the part in $\mathbb{G}1$ of the common reference strings of a GS proof system. Interestingly, this still allows the technique to work if the relation's witnesses to be extracted are only in $\mathbb{G}1$.

[0136] A similar problem may arise for the NIZK proof system for generating the signatures: it can be desirable to use zero-knowledge and strong derivation privacy for all the signatures released by honest platforms with subverted chips, and be able to extract (one single) forged signature during the unforgeability experiment. An embodiment may be unable to afford to have all the elements of the witness be group elements in $\mathbb{G}1$. For this reason, disclosed are knowledge of a commitment [t] 1 and of its opening [y]2. Unfortunately, this asymmetry cannot be broken when using Type-3 pairings.

[0137] To solve this problem a random oracle can be used together with ABO-NIZKs and the fact that, in the reduction, only one NIZK proof needs be extractable. An embodiment is configured to hash the signed message M (together with the basename bsn) using another random oracle H, and then use the label H (bsn,*M*) in the labeled NIZK. Abstractly, in the proof of security, by programming the random oracle H, the embodiment lifts the selective security offered by the ABO-NIZK systems to obtain adaptive security. For the disclosed instantiation based on the external Diffie-Hellman assumption, the computational complexity of an ABO-NIZK prover (and verifier) may be only three exponentiations less efficient than a not-labelled GS-NIZK, while the size of the NIZK proofs remains the same.

[0138] Building Blocks. In an embodiment, a scheme works over bilinear groups generated by a generator *G*, and it makes use of the following building blocks:

[0139] [First building block] A structure-preserving signature scheme *SS* = (KGensp, Sig_{sp}, Versp) where messages are elements of $\mathbb{G}1$ and signatures are in $\mathbb{G}_1^{\ell_1} \times \mathbb{G}_2^{\ell_2}$.

[0140] [Second building block] An ABO label-based re-randomizable NIZK Π_{sign} with label space being $\{0,1\}^{\lambda}$ and for the relationship *R*sign defined as:

$$\left\{ \begin{pmatrix} (\mathrm{gpk}, [\mathbf{b}]_1, \mathrm{SigRL}), \\ ([\mathrm{t}]_1, \sigma_{sp}, [\mathrm{y}]_2) \end{pmatrix} : \begin{array}{l} [\mathbf{b}]_1 \in span([1, y_0]_1^{\top}) \\ \forall i: [\mathbf{b}_i]_1 \notin span([1, y_0]_1^{\top}) \\ [t]_t = [\mathrm{h}^{\top} \cdot \mathrm{y}]_t \end{array} \right\}$$

$$\mathrm{Ver}_{sp}(\mathrm{pk}_{sp}, [t]_1, \sigma_{sp}) = 1$$

[0141] In the above expression, $\mathrm{SigRL} = \{[\mathbf{b}_i]_1\}_{i=1}^{r}$, gpk = ($[\mathbf{h}]_1$, pk_{sp}), and y = $(y_0, y_1)^{\top}$. To simplify the exposition, the description of the protocol below omits gpk (the public key of the scheme) from the instance and considers ($[\mathbf{b}]_1$, SigRL) as an instance for the relation.

[0142] [Third building block] A malleable and re-randomizable GS-NIZK

$$\mathcal{NIZK}_{\mathrm{com}}$$

for the following relationship

$$\mathcal{R}_{\mathrm{com}}$$

and set of transformations

$$\mathcal{T}_{\mathrm{com}}$$

defined below:

$$\{([\mathbf{h}]2,[t]2),[\mathbf{y}]1: e([1]1,[t]2) = e([\mathbf{y}]1,[\mathbf{h}]2)\}$$

$$\left\{T=(T_x,T_w): \begin{matrix} Tx([\mathbf{h}]2,[t]2) = [\mathbf{h}]2,[t+h_2\cdot y\,]_2 \\ T_w([\mathbf{y}]1) = [y0, y1\ +\ y\,]_1^{\top} \end{matrix}\right\}$$

[0143] Namely, the relation proves the knowledge (in $\mathbb{G}1$) of the opening of a Pedersen's commitment (in $\mathbb{G}2$) whose commitment key is [h]2. The transformation allows to re-randomize the commitment by adding fresh randomness. The common reference string of the NIZK is a vector in $\mathbb{G}_1^{\ell} \times \mathbb{G}_2^{\ell}$.

[0144] [Fourth building block] A

$$\mathcal{NIZK}_{\mathrm{hvt}}$$

for the relation

$$\mathcal{R}_{\mathrm{hvt}} = \{[x, xy, z, zy]1, y :$$

$$x, y, z \in \mathbb{Z}p\}.$$

[0145] [Fifth building block] Three cryptographic hash functions H, J and K modeled as random oracles, where H : {0, 1}* →- {0, 1}$^{\lambda}$, J : $\{0,1\}^* \rightarrow \mathbb{G}_1^{\ell}$ and K : {0,1}$\lambda \rightarrow \mathbb{G}1$, and the value $\ell \in \mathbb{N}$ depends only on the NIZK

$$\mathcal{NIZK}_{\mathrm{com}}.$$

[0146] The following describes an embodiment of a SR-EPID scheme:

[0147] Init(1^{λ}) →- pub: Generate description of a type-3 bilinear group bgp $\mathrm{bgp} \xleftarrow{\$} \mathcal{G}(1^{\lambda})$, the common reference strings $\mathrm{crs}_{\mathrm{sign}} \xleftarrow{\$} \mathcal{NIZK}_{\mathrm{sign}}$. Init(bgp),

$$\mathrm{crs}_{\mathrm{com},2}, \mathrm{tp}_{\mathrm{s}}$$

$$\xleftarrow{\$} \mathcal{NIZK}_{\mathrm{sign}}.$$

$\overline{\mathrm{Init}}_2$ (bgp), and $\mathrm{crs}_{\mathrm{hvt}} \xleftarrow{\$} \mathcal{NIZK}_{\mathrm{sign}}$. Init(bgp), and sample $\mathbf{h} \xleftarrow{\$} \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{2}}$. Output pub = (bgp, $\mathrm{crs}_{\mathrm{sign}}$, $\mathrm{crs}_{\mathrm{com},2}$, $\mathrm{crs}_{\mathrm{hvt}}$, $[\mathbf{h}]_1$, $[\mathbf{h}]_2$).

[0148] Setup(pub) →- (gpk, isk): sample $(\mathrm{sk}_{sp}, \mathrm{pk}_{sp}) \xleftarrow{\$} \mathrm{KGen}_{sp}(\mathrm{bgp})$, and set isk : = sk_{sp}, gpk := pk_{sp}. $\mathrm{Join}_{\mathcal{I},\mathcal{H},\mathcal{M}}\langle(\mathrm{gpk},\mathrm{isk}),\mathrm{gpk},\mathrm{gpk}\rangle \rightarrow \langle b,(b,\mathrm{hvt}),(\mathrm{sk},\mathrm{hvt})\rangle$: the platform $\mathcal{P}$ = ($\mathcal{M}$, $\mathcal{H}$) and issuer $\mathcal{I}$ start an interactive protocol that proceeds as described below:

1. $\mathcal{I}$ samples $id \xleftarrow{\$} \{0,1\}^{\lambda}$ and send id to $\mathcal{H}$ and $\mathcal{M}$. All the parties compute $crs_{com,1} \leftarrow J(id)$ and set $crs_{com} := (crs_{com,1}, crs_{com,2})$.

2. $\mathcal{H}$ samples y_0, $c \xleftarrow{\$} \mathbb{Z}_p$, set hvt := $[c, cy_0]_1$ and sends (y_0, hvt) to $\mathcal{M}$.

3. $\mathcal{M}$ does as described below:

- Sample $y_{\mathcal{M}} \xleftarrow{\$} \mathbb{Z}_p$ and compute $[t_{\mathcal{M}}]_1 := (y_0, y_{\mathcal{M}}) \cdot [\mathbf{h}]_1$ and $[t_{\mathcal{M}}]_2 := (y_0, y_{\mathcal{M}}) \cdot [\mathbf{h}]_2$;
- $\pi_{\mathcal{M}} \leftarrow \mathcal{NIZK}_{\mathrm{com}}.\mathrm{P}(\mathrm{crs}_{\mathrm{com}}, ([\mathbf{h}]_2, [t_{\mathcal{M}}]_2), [y_0, y_{\mathcal{M}}]_1)$;
- Send $([t_{\mathcal{M}}]_1, [t_{\mathcal{M}}]_2, \pi_{\mathcal{M}})$ to $\mathcal{H}$.

4. $\mathcal{H}$ checks $\mathcal{NIZK}_{\mathrm{com}}.\mathrm{V}(\mathrm{crs}_{\mathrm{com}}, ([\mathbf{h}]_2, [t_{\mathcal{M}}]_2, \pi_{\mathcal{M}}) = 1$ and $e([t_{\mathcal{M}}]_1, [1]_2) = e([1]_1, [t_{\mathcal{M}}]_2)$; if both checks pass:

- Sample $y_{\mathcal{H}} \xleftarrow{\$} \mathbb{Z}_p$ and set $[t]_1 := [t_{\mathcal{M}} + h_2 \cdot y_{\mathcal{H}}]_1$, $[t]_2 := [t_{\mathcal{M}} + h_2 \cdot y_{\mathcal{H}}]_2$;
- Compute $\pi_{\mathcal{H}} \leftarrow \mathcal{NIZK}_{\mathrm{com}}.\mathrm{ZKEval}(\mathrm{crs}_{\mathrm{com}}, \pi_{\mathcal{M}}, [y_{\mathcal{H}}]_1)$;
- Send $y_{\mathcal{H}}$ to $\mathcal{M}$ and $([t]_1, [t]_2, \pi_{\mathcal{H}})$ to $\mathcal{I}$.

5. $\mathcal{I}$ checks $\mathcal{NIZK}_{\mathrm{com}}.\mathrm{V}(\mathrm{crs}_{\mathrm{com}}, ([\mathbf{h}]_2, [t]_2), \pi_{\mathcal{H}}) = 1$ and $e([t]_1, [1]_2) = e([1]_1, [t]_2)$, and if the check passes then $\mathcal{I}$ computes $\sigma_{sp} \leftarrow \mathrm{Sig}_{sp}(\mathrm{sk}_{sp}, [t]_1)$ and sends σ_{sp} to $\mathcal{M}$ (through $\mathcal{H}$).

6. $\mathcal{M}$ does as described below:

- Compute $y_1 = y_{\mathcal{M}} + y_{\mathcal{H}}$ and set $\mathbf{y} := (y_0, y_1)^{\top}$;
- Verify (1) $[\mathbf{h}]_1^{\top} \cdot \mathbf{y} = [t]_1$ and (2) $\mathrm{Ver}_{sp}(\mathrm{pk}_{sp}, [t]_1, \sigma_{sp}) = 1$
- If so, send the special message completed to $\mathcal{I}$ (through $\mathcal{H}$) and output sk := ($[t]_1$, σ_{sp}, y) and hvt.

7. $\mathcal{H}$ outputs hvt.

8. If $\mathcal{I}$ receives the special message completed then outputs it.

[0149] Sig(gpk, sk, hvt, bsn, M, SigRL) $\rightarrow$- (σ, π_σ): On input gpk, sk = ($[t]_1$, σ_{sp}, y), the base name bsn $\in \{0,1\}^{\lambda}$, the message M E $\{0,1\}^m$, and a signature revocation list SigRL = $\{(\mathrm{bsn}_i, \mathrm{M}i, \sigma_i)\}_{i \in [n]}$), generate a signature σ and a proof π_σ as follows:

1. Set $[c]_1 \leftarrow K(\mathrm{bsn})$and set $[c]_1 := [c, c \cdot y_0]_1$;

2. Compute: $\pi \leftarrow \Pi_{\mathrm{sign}}.\mathrm{P}$ ($\mathrm{crs}_{\mathrm{sign}}$, H(bsn, M), ($[\mathbf{c}]_1$, SigRL), ([t]$_1$, $[\sigma_{sp}]_1$,$[y]_2$)) (the label is H(bsn, M))

3. Compute $\pi_\sigma \leftarrow \Pi_{\mathrm{hvt}}.\mathrm{P}(\mathrm{crs}_{\mathrm{hvt}}, (\mathrm{hvt}, [\mathbf{c}]_1), y_0)$;

4. Output $\sigma := ([\mathbf{c}]_1, \pi)$ and π_σ.

[0150] Sanitize(gpk, bsn, M, (σ, π_σ), SigRL, hvt): Parse $\sigma = ([\mathbf{c}]_1, \pi)$ and proceed as follows:

1. If

$$\Pi_{\mathrm{sign}}.\mathrm{V}\left(\mathrm{crs}_{\mathrm{sign}}, \mathrm{H}(\mathrm{bsn}, M), ([\mathbf{c}]_{\mathbf{1}}, \mathrm{SigRL}), \pi\right) = 0 \text{ or } \Pi_{\mathrm{hvt}}.\mathrm{V}(\mathrm{crs}_{\mathrm{hvt}}, (\mathrm{hvt}, [\mathbf{c}]_{\mathbf{1}}), \pi_{\sigma}) = 0 \text{ then output } \perp.$$

2. Re-randomize π by computing $\pi' \leftarrow \Pi_{\mathrm{sign}}$. ZKEval(crs, H(bsn, M), ($[\mathbf{c}]_1$, SigRL), π)

3. Output σ':= ($[\mathbf{c}]$,π').

[0151] Ver(gpk, bsn, *M*, σ, PrivRL, SigRL): Parse σ = ($[\mathbf{c}]_1$, π) and PrivRL := {f_1, ..., f_{n_1}}. Return 1 if and only if

1. K(bsn) = $[c]_1$,

2. Π_{sign}. V($\mathrm{crs}_{\mathrm{sign}}$, H(bsn, *M*), ($[\mathbf{c}]_1$, SigRL), π) and

3. For $\forall$sk $\in$ PrivRL : let sk = ($[f]_1$, σ_{sp}, (y_0,y_1)) check (-y_0, 1) · $[\boldsymbol{c}]_1 \neq [0]_1$.

[0152] Link (gpk, bsn, M_1, σ_1, SigRL_1, M_2, σ_2, SigRL_2) →- 0/1. Parse σ_i = ($[\mathbf{c}_i]_1$, π_i) for i = 1, 2. Return 1 if and only if $[\boldsymbol{c}_1]_1 = [\boldsymbol{c}_2]_1$ and both signatures are valid, i. e., Ver(gpk, bsn, M_1, σ_1, 0, SigRL_1) = 1 and Ver(gpk, bsn, M_2, σ_2, 0, SigRL_2) = 1.

[0153] While embodiments of the invention have been illustrated and described in detail in the drawings and foregoing description, such illustration and description are to be considered illustrative or exemplary and not restrictive. It will be understood that changes and modifications may be made by those of ordinary skill within the scope of the following claims. In particular, the present invention covers further embodiments with any combination of features from different embodiments described above and below. Additionally, statements made herein characterizing the invention refer to an embodiment of the invention and not necessarily all embodiments.

[0154] The terms used in the claims should be construed to have the broadest reasonable interpretation consistent with the foregoing description. For example, the use of the article "a" or "the" in introducing an element should not be interpreted as being exclusive of a plurality of elements. Likewise, the recitation of "or" should be interpreted as being inclusive, such that the recitation of "A or B" is not exclusive of "A and B," unless it is clear from the context or the foregoing description that only one of A and B is intended. Further, the recitation of "at least one of A, B and C" should be interpreted as one or more of a group of elements consisting of A, B and C, and should not be interpreted as requiring at least one of each of the listed elements A, *B* and C, regardless of whether A, B and C are related as categories or otherwise. Moreover, the recitation of "A, B and/or C" or "at least one of A, B or C" should be interpreted as including any singular entity from the listed elements, e.g., A, any subset from the listed elements, e.g., A and B, or the entire list of elements A, B and C.

Claims

1. A computer-implemented method comprising:

receiving, by a host (110) that comprises an intermediate processing system (114) and a trusted execution environment (112), a request for remote attestation of a reference binary from a remote verifier (120),
deploying the reference binary in the trusted execution environment (112) based on the received request,
calculating, with the trusted execution environment (112) and in response to the remote attestation request, an original digital signature based on the reference binary;
receiving, by the intermediate processing system (114), an original message from the trusted execution environment (112), the original message comprising the original digital signature authored by the trusted execution environment (112);
computing, by the intermediate processing system (114), a proof of knowledge for the original digital signature;
modifying, by the intermediate processing system (114), the original message by replacing the original digital signature with the proof of knowledge; and
forwarding, by the intermediate processing system (114), the modified original message comprising the proof of knowledge to the remote verifier (120).

2. The method of claim 1, wherein the original digital signature comprises an original revocation token and the modifying

of the original message comprises replacing the original revocation token with a randomized revocation token, and wherein the modified original message forwarded to the remote verifier (120) comprises the proof of knowledge and the randomized revocation token.

3. The method of claim 1 or 2, wherein the original message comprises a destination address and the method comprises transmitting the modified message to the destination address.

4. The method of any of claims 2 to 3, wherein the original revocation token is randomized by modifying one or more fields of the original revocation token based on a randomly selected parameter.

5. The method of any of claims 2 to 4, wherein the trusted execution environment (112) locally stores a trusted private key, the trusted execution environment (112) authors the original digital signature with the trusted private key, and the randomized revocation token includes parameters sufficient to revoke the trusted private key.

6. The method of any of claims 1 to 5, wherein the original message comprises a digest computed by the trusted execution environment (112) based on the reference binary.

7. The method of claim 6, wherein the digest comprises a hash of the reference binary.

8. A processing system comprising an intermediate processing system (114) and a trusted execution environment (112), each of which comprising one or more processors configured to:

receive a request for remote attestation of a reference binary from a remote verifier (120), deploy the reference binary in the trusted execution environment (112) based on the received request, and calculate, with the trusted execution environment (112) and in response to the remote attestation request, an original digital signature based on the reference binary;
receive an original message from the trusted execution environment (112), the original message comprising the original digital signature authored by the trusted execution environment (112);
compute a proof of knowledge for the original digital signature;
modify the original message by replacing the original digital signature with the proof of knowledge; and
forward the modified original message comprising the proof of knowledge to the remote verifier (120).

9. The processing system of claim 8, wherein the original digital signature comprises an original revocation token and the one or more processors are configured to modify the original message by replacing the original revocation token with a randomized revocation token, and wherein the modified original message forwarded to the verifier (120) comprises the proof of knowledge and the randomized revocation token.

10. A non-transitory computer-readable medium comprising code for configuring one or more processors of at least one of an intermediate processing system (114) and a trusted execution environment (112) to:

receive a request for remote attestation of a reference binary from a remote verifier (120), deploy the reference binary in the trusted execution environment (112) based on the received request, and calculate, with the trusted execution environment (112) and in response to the remote attestation request, an original digital signature based on the reference binary;
receive an original message from the trusted execution environment (112), the original message comprising the original digital signature authored by the trusted execution environment(112);
compute a proof of knowledge for the original digital signature;
modify the original message by replacing the original digital signature with the proof of knowledge; and
forward the modified original message comprising the proof of knowledge to the remote verifier (120).

11. The non-transitory computer-readable medium of claim 10, wherein the original digital signature comprises an original revocation token, wherein the code for modifying the original message comprises code for configuring the one or more processors to replace the original revocation token with a randomized revocation token, and wherein the code for forwarding the modified original message to the remote verifier (120) comprises code for transmitting to the verifier (120) the modified original message comprising the proof of knowledge and the randomized revocation token.

Patentansprüche

1. Computer-implementiertes Verfahren, das Folgendes umfasst:

   Empfangen, durch einen Host (110), der ein Zwischenverarbeitungssystem (114) und eine vertrauenswürdige Ausführungsumgebung (112) umfasst, einer Anfrage zur Integritätsüberprüfung eines Referenz-Binärprogramms von einem Remote-Verifizierer (120),
   Bereitstellen des Referenz-Binärprogramms in der vertrauenswürdigen Ausführungsumgebung (112) basierend auf der empfangenen Anfrage,
   Berechnen, mit der vertrauenswürdigen Ausführungsumgebung (112) und in Erwiderung auf die Anfrage zur Integritätsüberprüfung, einer ursprünglichen digitalen Signatur auf der Grundlage des Referenz-Binärprogramms;
   Empfangen einer ursprünglichen Nachricht von der vertrauenswürdigen Ausführungsumgebung (112) durch das Zwischenverarbeitungssystem (114), wobei die ursprüngliche Nachricht die ursprüngliche digitale Signatur umfasst, die von der vertrauenswürdigen Ausführungsumgebung (112) erstellt wurde;
   Berechnen eines Wissensnachweises für die ursprüngliche digitale Signatur durch das Zwischenverarbeitungssystem (114);
   Modifizieren der ursprünglichen Nachricht durch das Zwischenverarbeitungssystem (114), indem die ursprüngliche digitale Signatur durch den Wissensnachweis ersetzt wird; und
   Weiterleiten der modifizierten Originalnachricht, die den Wissensnachweis enthält, durch das Zwischenverarbeitungssystem (114) an den Remote-Verifizierer (120).

2. Verfahren nach Anspruch 1, wobei die ursprüngliche digitale Signatur ein ursprüngliches Widerrufstoken umfasst und das Modifizieren der ursprünglichen Nachricht das Ersetzen des ursprünglichen Widerrufstokens durch ein randomisiertes Widerrufstoken umfasst, und wobei die modifizierte ursprüngliche Nachricht, die an den Remote-Verifizierer (120) weitergeleitet wird, den Wissensnachweis und das randomisierte Widerrufstoken umfasst.

3. Verfahren nach Anspruch 1 oder 2, wobei die ursprüngliche Nachricht eine Zieladresse umfasst und das Verfahren das Übertragen der modifizierten Nachricht an die Zieladresse umfasst.

4. Verfahren nach einem der Ansprüche 2 bis 3, wobei das ursprüngliche Widerrufstoken randomisiert wird, indem ein oder mehrere Felder des ursprünglichen Widerrufstokens auf der Grundlage eines zufällig ausgewählten Parameters modifiziert werden.

5. Verfahren nach einem der Ansprüche 2 bis 4, wobei die vertrauenswürdige Ausführungsumgebung (112) einen vertrauenswürdigen privaten Schlüssel lokal speichert, die vertrauenswürdige Ausführungsumgebung (112) die ursprüngliche digitale Signatur mit dem vertrauenswürdigen privaten Schlüssel erstellt und das randomisierte Widerrufstoken Parameter enthält, die ausreichen, um den vertrauenswürdigen privaten Schlüssel zu widerrufen.

6. Verfahren nach einem der Ansprüche 1 bis 5, wobei die ursprüngliche Nachricht einen von der vertrauenswürdigen Ausführungsumgebung (112) auf der Grundlage des Referenz-Binärprogramms berechneten Digest umfasst.

7. Verfahren nach Anspruch 6, wobei der Digest einen Hash des Referenz-Binärprogramms umfasst.

8. Verarbeitungssystem umfassend ein Zwischenverarbeitungssystem (114) und eine vertrauenswürdige Ausführungsumgebung (112), jeweils einen oder mehrere Prozessoren umfassend, die konfiguriert sind

   eine Anfrage zur Integritätsüberprüfung eines Referenz-Binärprogramms von einem Remote-Verifizierer (120) zu empfangen, das Referenz-Binärprogramm in der vertrauenswürdigen Ausführungsumgebung (112) basierend auf der empfangenen Anfrage bereitzustellen, und, mit der vertrauenswürdigen Ausführungsumgebung (112) und in Erwiderung auf die Anfrage zur Integritätsüberprüfung, eine ursprüngliche digitale Signatur auf der Grundlage des Referenz-Binärprogramms zu berechnen;
   eine ursprüngliche Nachricht von der vertrauenswürdigen Ausführungsumgebung (112) zu empfangen, wobei die ursprüngliche Nachricht die ursprüngliche digitale Signatur umfasst, die von der vertrauenswürdigen Ausführungsumgebung (112) erstellt wurde;
   einen Wissensnachweises für die ursprüngliche digitale Signatur zu berechnen;
   die ursprüngliche Nachricht zu modifizieren, indem die ursprüngliche digitale Signatur durch den Wissensnachweis ersetzt wird; und

die modifizierte Originalnachricht, die den Wissensnachweis enthält, an den Remote-Verifizierer (120) weiterzuleiten.

9. Verarbeitungssystem nach Anspruch 8, wobei die ursprüngliche digitale Signatur ein ursprüngliches Widerrufstoken umfasst und der eine oder die mehreren Prozessoren so konfiguriert sind, dass sie die ursprüngliche Nachricht modifizieren, indem sie das ursprüngliche Widerrufstoken durch ein randomisiertes Widerrufstoken ersetzen, und wobei die modifizierte ursprüngliche Nachricht, die an den Verifizierer (120) weitergeleitet wird, den Wissensnachweis und das randomisierte Widerrufstoken umfasst.

10. Nicht-transitorisches computerlesbares Medium, das Code zum Konfigurieren eines oder mehrerer Prozessoren von mindestens einem von einem Zwischenverarbeitungssystem (114) und einer vertrauenswürdigen Ausführungsumgebung (112) umfasst, um:

eine Anfrage zur Integritätsüberprüfung eines Referenz-Binärprogramms von einem Remote-Verifizierer (120) zu empfangen, das Referenz-Binärprogramm in der vertrauenswürdigen Ausführungsumgebung (112) basierend auf der empfangenen Anfrage bereitzustellen und, mit der vertrauenswürdigen Ausführungsumgebung (112) und in Erwiderung auf die Anfrage zur Integritätsüberprüfung eine ursprüngliche digitale Signatur auf der Grundlage des Referenz-Binärprogramms zu berechnen;
eine ursprüngliche Nachricht von der vertrauenswürdigen Ausführungsumgebung (112) zu empfangen, wobei die ursprüngliche Nachricht die ursprüngliche digitale Signatur umfasst, die von der vertrauenswürdigen Ausführungsumgebung (112) erstellt wurde;
einen Wissensnachweises für die ursprüngliche digitale Signatur zu berechnen;
die ursprüngliche Nachricht zu modifizieren, indem die ursprüngliche digitale Signatur durch den Wissensnachweis ersetzt wird; und
die modifizierte Originalnachricht, die den Wissensnachweis enthält, an den Remote-Verifizierer (120) weiterzuleiten.

11. Nicht-transitorisches computerlesbares Medium nach Anspruch 10, wobei die ursprüngliche digitale Signatur ein ursprüngliches Widerrufstoken umfasst, wobei der Code zum Modifizieren der ursprünglichen Nachricht einen Code zum Konfigurieren des einen oder der mehreren Prozessoren umfasst, um das ursprüngliche Widerrufstoken durch ein randomisiertes Widerrufstoken zu ersetzen, und wobei der Code zum Weiterleiten der modifizierten ursprünglichen Nachricht an den Remote-Verifizierer (120) einen Code zum Übertragen der modifizierten ursprünglichen Nachricht mit dem Wissensnachweis und dem randomisierten Widerrufstoken an den Verifizierer (120) umfasst.

Revendications

1. Procédé mis en oeuvre par ordinateur comprenant :

la réception, par un hôte (110) qui comprend un système de traitement intermédiaire (114) et un environnement d'exécution digne de confiance (112), d'une demande d'attestation distante d'un binaire de référence depuis un vérificateur distant (120),
le déploiement du binaire de référence dans l'environnement d'exécution digne de confiance (112) sur la base de la demande reçue,
le calcul, avec l'environnement d'exécution digne de confiance (112) et en réponse à la demande d'attestation distante, d'une signature numérique initiale sur la base du binaire de référence ;
la réception, par le système de traitement intermédiaire (114), d'un message initial depuis l'environnement d'exécution digne de confiance (112), le message initial comprenant la signature numérique initiale créée par l'environnement d'exécution digne de confiance (112) ;
le calcul, par le système de traitement intermédiaire (114), d'une preuve de connaissance pour la signature numérique initiale ;
la modification, par le système de traitement intermédiaire (114), du message initial par le remplacement de la signature numérique initiale par la preuve de connaissance ; et
l'expédition, par le système de traitement intermédiaire (114), du message initial modifié comprenant la preuve de connaissance au vérificateur distant (120).

2. Procédé selon la revendication 1, dans lequel la signature numérique initiale comprend un jeton de révocation initial et la modification du message initial comprend le remplacement du jeton de révocation initial par un jeton de

révocation randomisé, et dans lequel le message initial modifié expédié au vérificateur distant (120) comprend la preuve de connaissance et le jeton de révocation randomisé.

3. Procédé selon la revendication 1 ou 2, dans lequel le message initial comprend une adresse de destination et le procédé comprend la transmission du message modifié à l'adresse de destination.

4. Procédé selon la revendication 2 ou 3, dans lequel le jeton de révocation initial est randomisé par la modification d'un ou plusieurs champs du jeton de révocation initial sur la base d'un paramètre sélectionné aléatoirement.

5. Procédé selon l'une quelconque des revendications 2 à 4, dans lequel l'environnement d'exécution digne de confiance (112) stocke localement une clé privée digne de confiance, l'environnement d'exécution digne de confiance (112) crée la signature numérique initiale avec la clé privée digne de confiance, et le jeton de révocation randomisé comporte des paramètres suffisants pour révoquer la clé privée digne de confiance.

6. Procédé selon l'une quelconque des revendications 1 à 5, dans lequel le message initial comprend un condensé calculé par l'environnement d'exécution digne de confiance (112) sur la base du binaire de référence.

7. Procédé selon la revendication 6, dans lequel le condensé comprend un hachage du binaire de référence.

8. Système de traitement comprenant un système de traitement intermédiaire (114) et un environnement d'exécution digne de confiance (112), chacun d'eux comprenant un ou plusieurs processeurs configurés pour :

recevoir une demande d'attestation distante d'un binaire de référence depuis un vérificateur distant (120), déployer le binaire de référence dans l'environnement d'exécution digne de confiance (112) sur la base de la demande reçue, et calculer, avec l'environnement d'exécution digne de confiance (112) et en réponse à la demande d'attestation distante, une signature numérique initiale sur la base du binaire de référence ;
recevoir un message initial depuis l'environnement d'exécution digne de confiance (112), le message initial comprenant la signature numérique initiale créée par l'environnement d'exécution digne de confiance (112) ;
calculer une preuve de connaissance pour la signature numérique initiale ;
modifier le message initial par le remplacement de la signature numérique initiale par la preuve de connaissance ; et
expédier le message initial modifié comprenant la preuve de connaissance au vérificateur distant (120).

9. Système de traitement selon la revendication 8, dans lequel la signature numérique initiale comprend un jeton de révocation initial et les un ou plusieurs processeurs sont configurés pour modifier le message initial par le remplacement du jeton de révocation initial par un jeton de révocation randomisé, et dans lequel le message initial modifié expédié au vérificateur (120) comprend la preuve de connaissance et le jeton de révocation randomisé.

10. Support non transitoire lisible par ordinateur comprenant un code pour configurer un ou plusieurs processeurs d'au moins l'un parmi un système de traitement intermédiaire (114) et un environnement d'exécution digne de confiance (112) pour :

recevoir une demande d'attestation distante d'un binaire de référence depuis un vérificateur distant (120), déployer le binaire de référence dans l'environnement d'exécution digne de confiance (112) sur la base de la demande reçue, et calculer, avec l'environnement d'exécution digne de confiance (112) et en réponse à la demande d'attestation distante, une signature numérique initiale sur la base du binaire de référence ;
recevoir un message initial depuis l'environnement d'exécution digne de confiance (112), le message initial comprenant la signature numérique initiale créée par l'environnement d'exécution digne de confiance (112) ;
calculer une preuve de connaissance pour la signature numérique initiale ;
modifier le message initial par le remplacement de la signature numérique initiale par la preuve de connaissance ; et
expédier le message initial modifié comprenant la preuve de connaissance au vérificateur distant (120).

11. Support non transitoire lisible par ordinateur selon la revendication 10, dans lequel la signature numérique initiale comprend un jeton de révocation initial, dans lequel le code pour modifier le message initial comprend un code pour configurer les un ou plusieurs processeurs pour remplacer le jeton de révocation initial par un jeton de révocation randomisé, et dans lequel le code pour expédier le message initial modifié au vérificateur (120) distant comprend un code pour transmettre, au vérificateur (120), le message initial modifié comprenant la preuve de connaissance et le jeton de révocation randomisé.

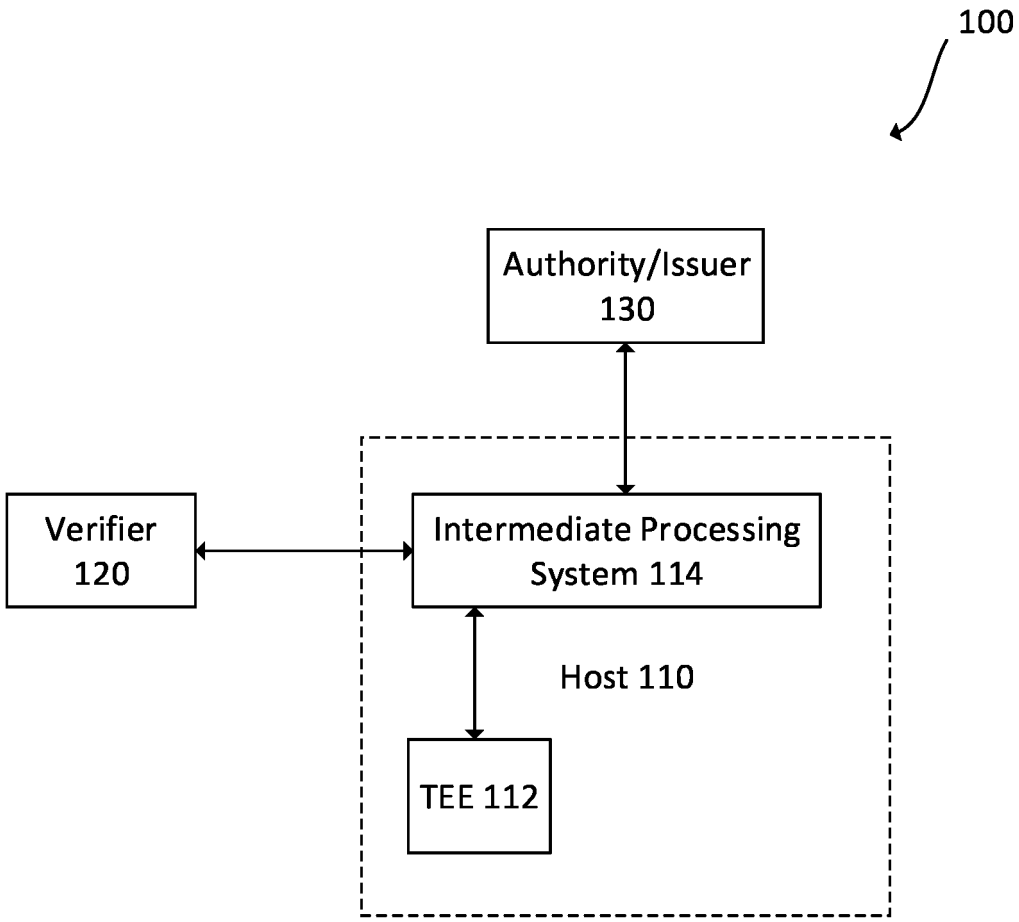
100
Authority/Issuer
130
Verifier
120
Intermediate Processing
System 114
Host 110
TEE 112

FIG. 1

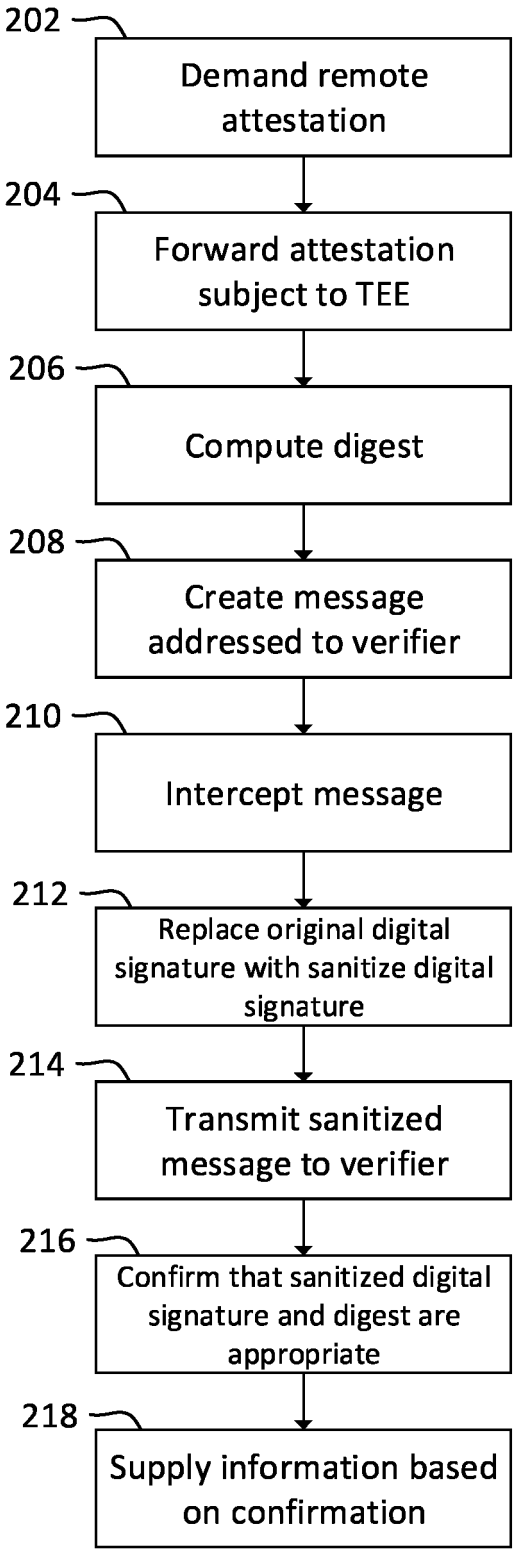
202
Demand remote attestation
204
Forward attestation subject to TEE
206
Compute digest
208
Create message addressed to verifier
210
Intercept message
212
Replace original digital signature with sanitize digital signature
214
Transmit sanitized message to verifier
216
Confirm that sanitized digital signature and digest are appropriate
218
Supply information based on confirmation

FIG. 2

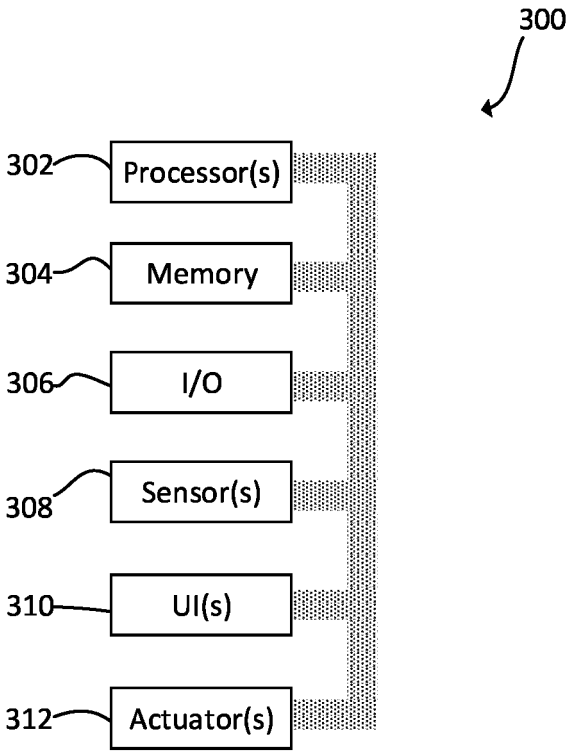
300
302
Processor(s)
304
Memory
306
I/O
308
Sensor(s)
310
UI(s)
312
Actuator(s)

FIG. 3

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- WO 2017049111 A1 **[0007]**

Non-patent literature cited in the description

- **GIUSEPPE ATENIESE ; BERNARDO MAGRI ; DANIELE VENTURI.** Subversion-Resilient Signature Schemes. *22nd ACM Conference on Computer and Communications Security,* 2015, vol. 2015, 364-375 **[0005]**
- **J. CAMENISCH et al.** Anonymous Attestation with Subverted TPMs. *IACR, International Association for Cryptologic Research,* 28 June 2017, vol. 20170628 (145434), 1-79 **[0006]**
- **CLAUS SCHNORR.** Efficient Identification and Signatures for Smart Cards. *Journal of Cryptology,* 1991, vol. 4 (3 **[0058]**
- **DAVID CHAUM ; JAN-HENDRIK EVERTSE ; JEROEN VAN DE GRAAF.** An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. *EUROCRYPT,* 1987 **[0059]**
- **DAVID CHAUM ; TORBEN PEDERSEN.** Wallet Databases with Observers. *CRYPTO,* vol. 1 **[0060]**
- **JAN CAMENISCH ; VICTOR SHOUP.** Practical Verifiable Encryption and Decryption of Discrete Logarithms. *CRYPTO,* 2003 **[0062]**
- **JENS GROTH ; AMIT SAHAI.** Efficient Non-interactive Proof Systems for Bilinear Groups. *IACR Cryptology ePrint Archive,* 2007, vol. 155 **[0065]**
- **MAN HO AU ; WILLY SUSILO ; YI MU.** Constant-Size Dynamic k-TAA. *SCN,* 2006 **[0067]**
- **ERNIE BRICKELL ; JIANGTAO LI.** Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation. *SocialCom/PASSAT,* 2010 **[0085]**